main.py

Save Run Output

```python
def two_sum(nums, target):
    num_indices = {}
    for i, num in enumerate(nums):
        complement = target - num
        if complement in num_indices:
            return [num_indices[complement], i]
        num_indices[num] = i
nums = [2, 7, 11, 15]
target = 9
print(two_sum(nums, target))
```

```
[0, 1]

=== Code Execution Successful ===
```

main.py

Save Run Output Clear

```
1 def isPalindrome(x):
2     x_str = str(x)
3     return x_str == x_str[::-1]
4 print(isPalindrome(121))
5 print(isPalindrome(-121))
6 print(isPalindrome(10))
7
```

```
True
False
False

=== Code Execution Successful ===
```

main.py

Save | Run | Output

```python
def myAtoi(s):
    INT_MAX = 2**31 - 1
    INT_MIN = -2**31
    s = s.lstrip()
    if not s:
        return 0
    sign = 1
    if s[0] == '-':
        sign = -1
        s = s[1:]
    elif s[0] == '+':
        s = s[1:]
    result = 0
    for char in s:
        if char.isdigit():
            result = result * 10 + int(char)
        else:
            break
    result *= sign
    if result < INT_MIN:
        return INT_MIN
    elif result > INT_MAX:
        return INT_MAX
    else:
        return result
print(myAtoi("42"))
```

```
42
4193
-2147483648

=== Code Execution Successful ===
```

main.py

Save | Run | Output | Clear

```python
def reverse(x):
    INT_MAX = 2**31 - 1
    INT_MIN = -2**31
    reversed_x = 0
    if x < 0:
        sign = -1
        x = abs(x)
    else:
        sign = 1
    while x != 0:
        digit = x % 10
        if reversed_x > (INT_MAX - digit) // 10:
            return 0
        reversed_x = reversed_x * 10 + digit
        x //= 10

    return sign * reversed_x
x = 123
print(reverse(x))
x = 120
print(reverse(x))
x = 0
print(reverse(x))

```

```
321
21
0

=== Code Execution Successful ===
```

main.py

Save Run

```python
def convert(s, numRows):
    if numRows == 1 or numRows >= len(s):
        return s
    rows = [''] * numRows
    current_row, direction = 0, 1
    for char in s:
        rows[current_row] += char
        current_row += direction
        if current_row == 0 or current_row == numRows - 1:
            direction *= -1
    return ''.join(rows)
s = "PAYPALISHIRING"
numRows = 3
print(convert(s, numRows))
```

Output Clear

```
PAHNAPLSIIGYIR

=== Code Execution Successful ===
```

main.py

Save | Run | Output | Clear

```python
def longestPalindrome(s):
    n = len(s)
    if n == 0:
        return ""
    dp = [[False] * n for _ in range(n)]
    start, max_length = 0, 1
    for i in range(n):
        dp[i][i] = True
    for i in range(n - 1):
        if s[i] == s[i + 1]:
            dp[i][i + 1] = True
            start = i
            max_length = 2
    for length in range(3, n + 1):
        for i in range(n - length + 1):
            j = i + length - 1
            if s[i] == s[j] and dp[i + 1][j - 1]:
                dp[i][j] = True
                start = i
                max_length = length
    return s[start:start + max_length]
print(longestPalindrome("babad"))
print(longestPalindrome("cbbd"))

```

```
aba
bb

=== Code Execution Successful ===
```

main.py

Save Run Output Clear

```python
def findMedianSortedArrays(nums1, nums2):
    nums = sorted(nums1 + nums2)
    n = len(nums)
    if n % 2 == 0:
        return (nums[n // 2 - 1] + nums[n // 2]) / 2
    else:
        return nums[n // 2]

```

```
=== Code Execution Successful ===
```

main.py

Save Run Output Clear

```python
def lengthOfLongestSubstring(s):
    char_index = {}
    max_length = 0
    start = 0

    for end, char in enumerate(s):
        if char in char_index and char_index[char] >= start:
            start = char_index[char] + 1
        else:
            max_length = max(max_length, end - start + 1)
        char_index[char] = end

    return max_length
print(lengthOfLongestSubstring("abcabcbb"))
```

```
3

=== Code Execution Successful ===
```

main.py

Save Run Output

```python
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next

def addTwoNumbers(l1, l2):
    dummy_head = ListNode(0)
    current = dummy_head
    carry = 0

    while l1 or l2 or carry:
        sum_val = carry

        if l1:
            sum_val += l1.val
            l1 = l1.next
        if l2:
            sum_val += l2.val
            l2 = l2.next

        carry = sum_val // 10
        current.next = ListNode(sum_val % 10)
        current = current.next

    return dummy_head.next
```

```
7 0 8
=== Code Execution Successful ===
```

main.py

Save | Run | Output | Clear

```python
def isMatch(s, p):
    dp = [[False] * (len(p) + 1) for _ in range(len(s) + 1)]
    dp[0][0] = True
    for j in range(1, len(p) + 1):
        if p[j - 1] == '*':
            dp[0][j] = dp[0][j - 2]
    for i in range(1, len(s) + 1):
        for j in range(1, len(p) + 1):
            if p[j - 1] == '.' or p[j - 1] == s[i - 1]:
                dp[i][j] = dp[i - 1][j - 1]
            elif p[j - 1] == '*':
                dp[i][j] = dp[i][j - 2] or (dp[i - 1][j] and (s[i - 1] == p[j -
                    2] or p[j - 2] == '.'))
    return dp[len(s)][len(p)]
print(isMatch("aa", "a"))
print(isMatch("aa", "a*"))
print(isMatch("ab", ".*"))
print(isMatch("mississippi", "mis*is*p*."))

```

```
False
True
True
False

=== Code Execution Successful ===
```